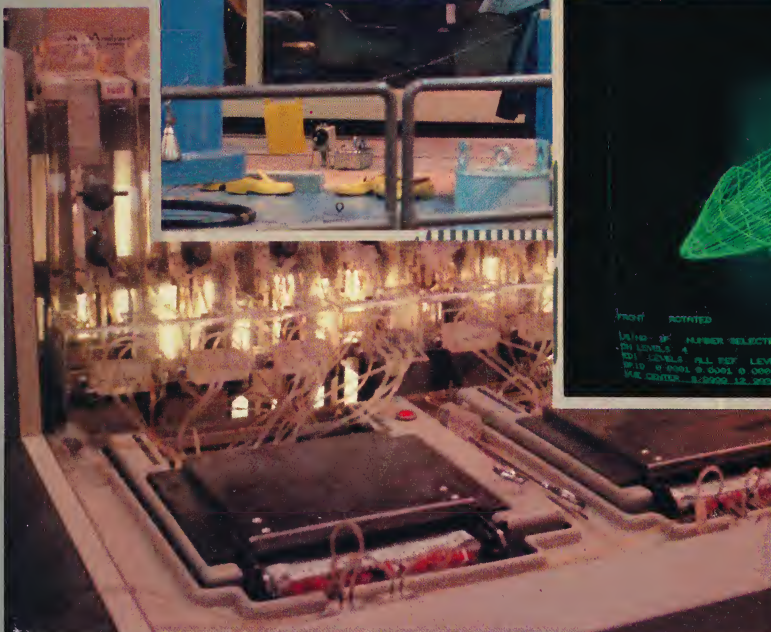
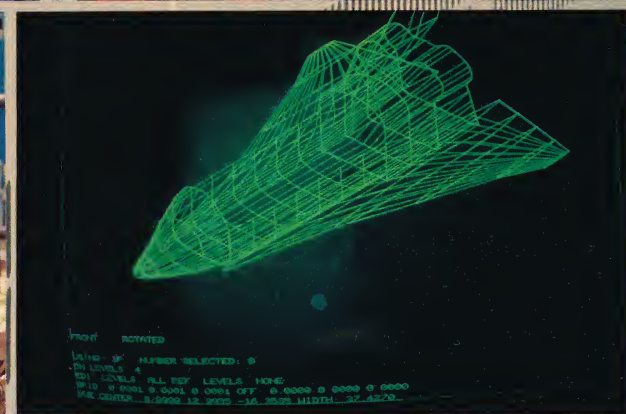
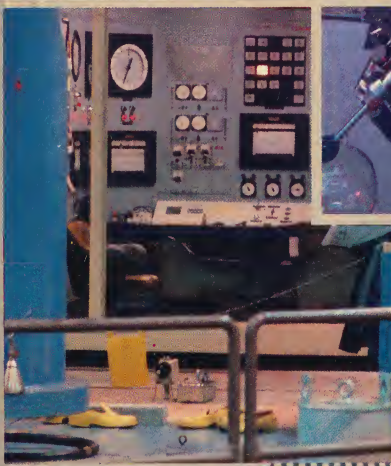
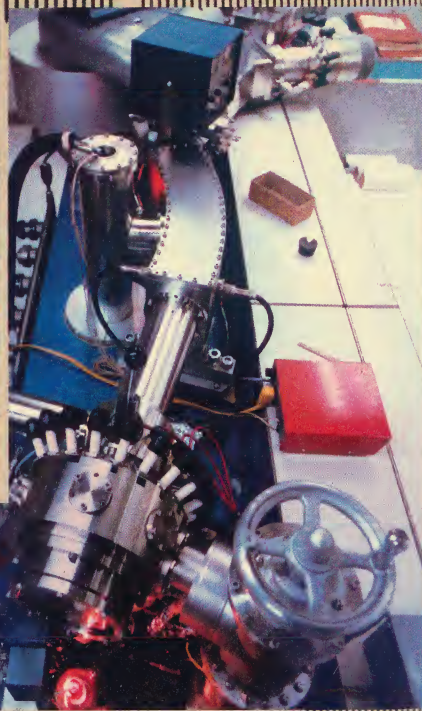


VAX Real-time Capabilities

digital VAX 11/780



digital

DIGITAL'S EXPERIENCE IN REAL-TIME PROCESSING	2
THE VAX ARCHITECTURE IN A REAL-TIME ENVIRONMENT	5
VAX REAL-TIME INTERFACES	11
VAX APPLICATION DEVELOPMENT TOOLS	14
VAX REAL-TIME APPLICATIONS	18

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear.

VAX, VMS, SBI, PDP, UNIBUS, MASSBUS are trademarks of Digital Equipment Corporation

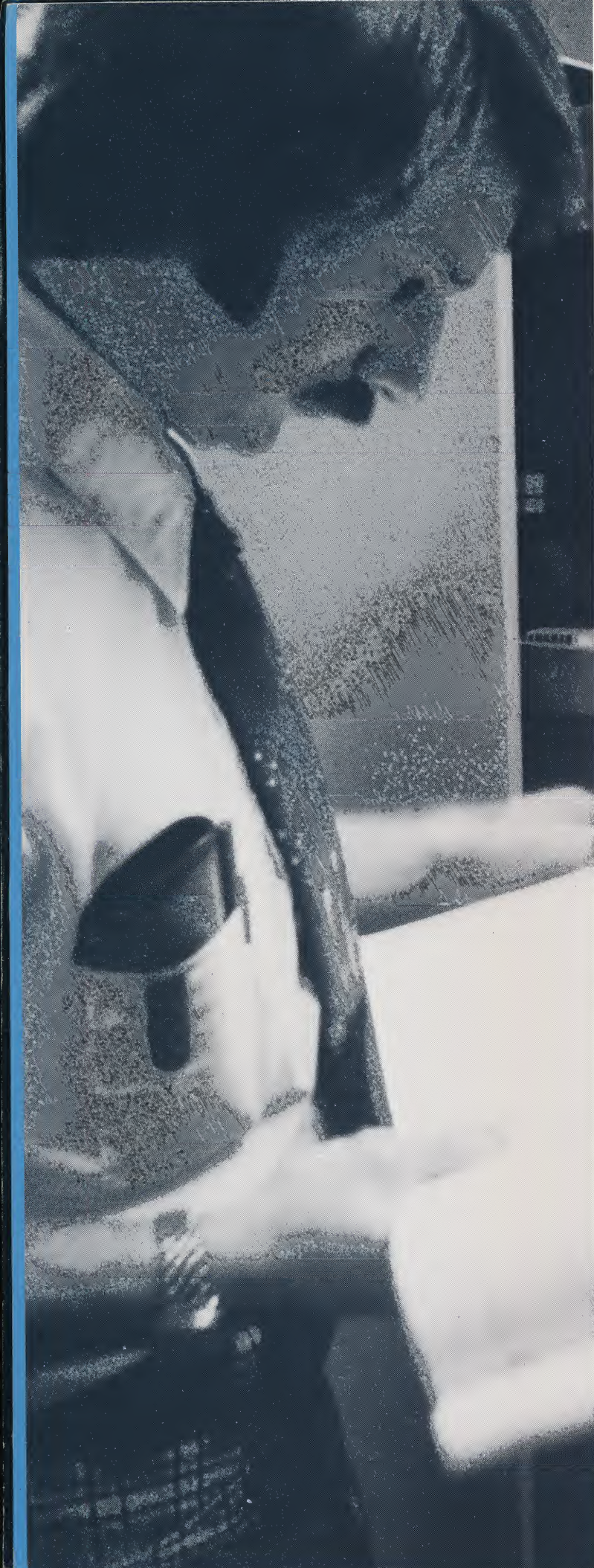
Copyright ©1979, by Digital Equipment Corporation

Sep 80

Anyone who is considering acquiring a computer system to solve real-time data acquisition and computational problems wants to ask some basic questions:

- What experience does the manufacturer have in building computers for use in a real-time environment?
- What makes the computer architecture especially appropriate for real-time applications?
- What interfaces are available for connecting a real-time process to the system?
- What system software tools are there for acquiring, manipulating, and analyzing data?
- How has the computer system already been used in the real-time environment?

DIGITAL and the VAX-11/780 have the answers.



Digital's Experience in Real-time Processing

As the world's leading manufacturer of minicomputers, DIGITAL has always identified very closely with the real-time user community.

The corporation's foundations were built on state-of-the-art technology when, in 1957, its founders first set up shop to manufacture electronic modules—functional building blocks for use by experimenters and researchers.

In the early 1960's, DIGITAL developed a series of modules for Lincoln Labs to use in the world's first Laboratory Instrument Computer—the LINC. The LINC was the first interactive computer designed to acquire and analyze live data in a real-time environment. It introduced many of the elements which today are taken for granted in real-time operation—an analog-to-digital converter, an interactive display screen, magnetic tape for mass data storage, and a main memory for program and temporary data storage.

In 1965, DIGITAL introduced the well known PDP-8 and then combined this general-purpose minicomputer with the special-purpose LINC to create the most powerful real-time computer of its time, the LINC-8. The LINC-8 achieved astounding success among scientists and researchers; its use as a research tool encouraged the formation of many new research programs.

Other real-time enhancements soon followed: the LAB-8, a low cost laboratory package, and IDACS-8, a real-time industrial application package. New real-time interfaces and the continued development of real-

time software packages proved the PDP-8 family's acceptance for scientific and industrial applications.

In 1969, DIGITAL developed the PDP-12, successor to the LINC-8. The PDP-12 combined the PDP-8's instruction set with a second instruction set designed for real-time I/O. The PDP-12 included more precise real-time interfaces, a new and larger interactive display screen, and a wider selection of special-purpose real-time application packages.

In spite of heavy user investment in the 12-bit architecture, substantial pressure for increased memory addressing capabilities caused DIGITAL to introduce, in the early 1970's, a new generation and a new family of real-time computers—the 16-bit PDP-11 family. This family's increased addressing resources, coupled with decreasing memory prices, presented new and exciting possibilities for real-time applications.

In addition, in keeping with the practice of providing application tools and modules for real-time users, DIGITAL introduced a line of I/O interfaces, hardware, and development tools for the new PDP-11 family, allowing users to design their own application systems.

At approximately this time, DIGITAL also began the formation of marketing groups to service the needs of a growing user community and to study the requirements of specific marketplaces.

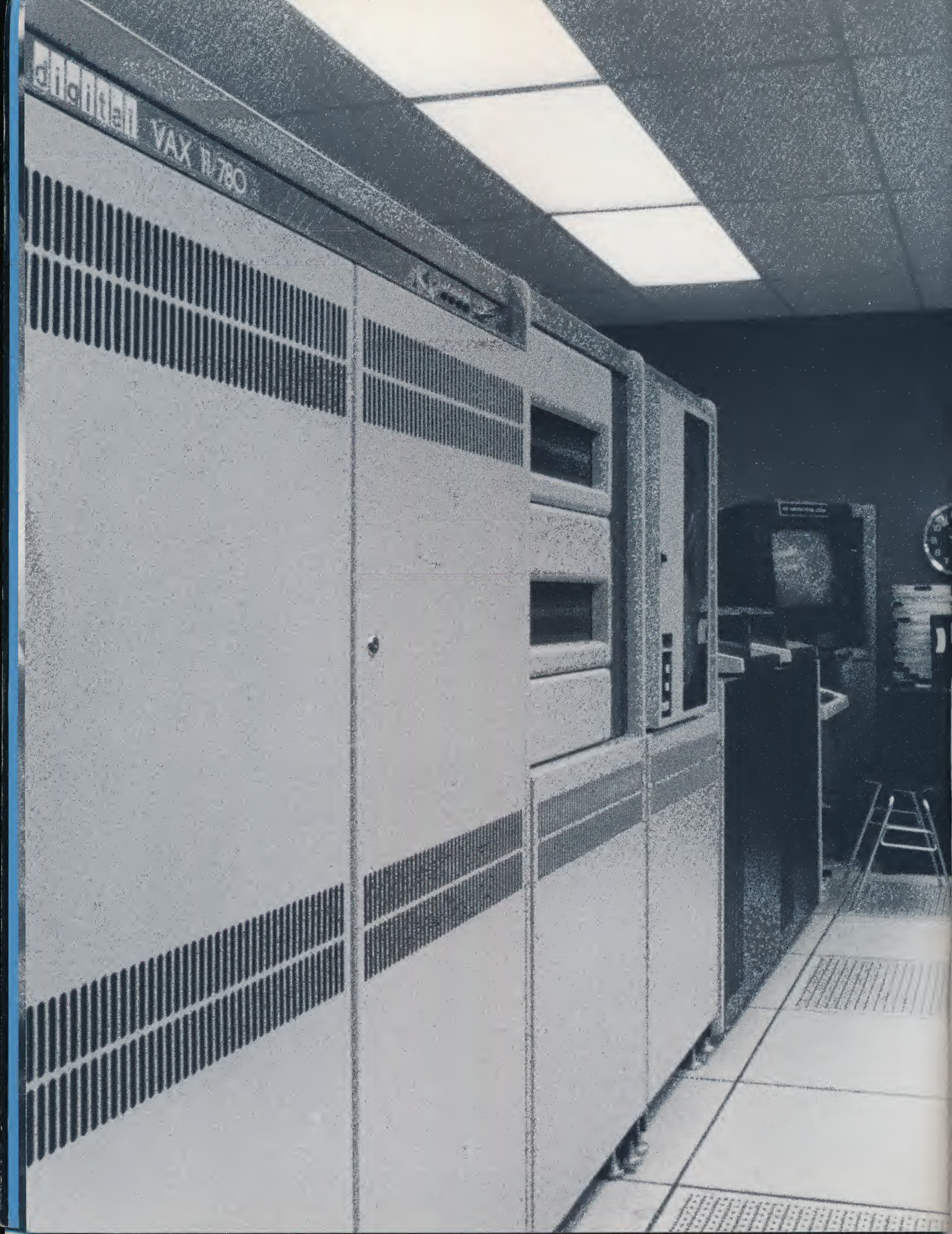
These marketing groups insured continued focus on the needs of real-time users. Special-purpose real-time application tools were developed for use in data analysis, sig-

nal processing, engineering design, graphics, industrial control, medical research, and clinical evaluation.

Examples of the new marketing orientation included such tools as ICS/ICR, an industrial control subsystem for local and remote use; IDACS-11, a real-time industrial application package; the Laboratory Peripheral Subsystem (LPS-11), a real-time front end providing connections for real-time data acquisition and output; and the DECLABs, a series of computer systems that combined computational power with standard real-time interfaces.

Now, DIGITAL's VAX-11 architecture represents another step forward in real-time computing. The VAX-11/780—the first implementation of this new 32-bit architecture—is the result of a dedicated design effort at DIGITAL, driven by hardware and software engineers with years of real-time experience. In fact, many of the people involved in the development of DIGITAL's earlier real-time offerings were directly responsible for the design of the VAX-11 architecture.

The VAX-11/780 is a 32-bit computer offering unlimited addressing capabilities for large user programs, a fast and efficient scheduler, a multi-level priority structure, direct real-time device support, easy-to-use development tools, and virtual memory. The VAX-11's advanced hardware and software design, coupled with microprocessor front-ends for data acquisition, create an unbeatable combination for today's sophisticated real-time user.



The VAX Architecture in a Real-time Environment

“Real-time” means different things to different users.

To some users, it means being able to acquire data at a very high rate—several thousands of samples per second, or even hundreds of thousands of samples per second. To other users, real-time means fast response to an external event. To still other users, real-time means being able to monitor input signals and provide output signals to control an on-line process.

Some users consider real-time as the ability to make a number of complex decisions in a short space of time. Some think of it as the ability to perform concurrent data acquisition and analysis. Still others define real-time as simply the ability to obtain immediate feedback to inquiries entered at a computer terminal.

Regardless of definition, the real-time capability of a computer system can be measured in terms of three fundamental parameters:

- **System Throughput:** Number of samples, i.e., data transfers, that can occur between an external device and the computer system in a given unit of time.
- **System Responsiveness:** Elapsed time from the occurrence of an external stimulus to the response given by the computer system.
- **System Processing Power:** Number of decisions that the computer system can perform per event.

The VAX-11 architecture was designed to facilitate all three—through its wide I/O bandwidth and real-time interfaces, its in-

terrupt responsiveness, and its powerful 32-bit processing capabilities.

VAX System Throughput

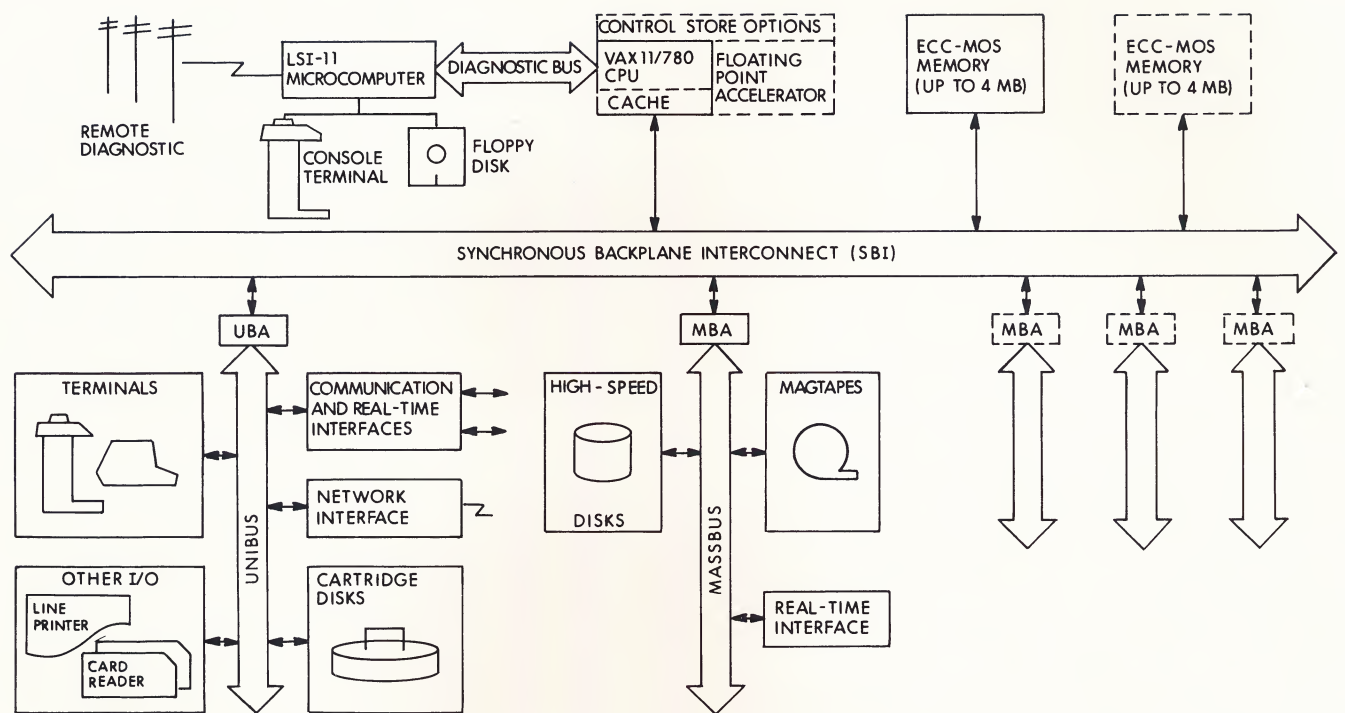
System throughput is measured by the number of data transfers that can occur in a given unit of time. System throughput is determined by the capability of system buses and real-time devices to transfer large amounts of information rapidly. Special operating system support also can be an influencing factor.

VAX-11/780 Bus Bandwidths: The VAX-11/780 uses an internal 32-bit wide backplane, called the Synchronous Backplane Interconnect (SBI), as the central path in its I/O system. The SBI handles all information transfers between the VAX-11 processor and memory, and between memory and device controllers.

The VAX-11/780's SBI has a cycle time of 200 nanoseconds. It can transfer a full 32 bits of data on each cycle, and thereby provides a total data transfer rate of 13.3 million bytes per second.

Two buses physically connect I/O devices to a VAX-11/780 system: the UNIBUS and the MASSBUS.

The UNIBUS is DIGITAL's standard interface for devices with high-speed, direct memory access transfer requirements. The UNIBUS adapter (UBA) links the UNIBUS to the SBI and maps UNIBUS address and data information to appropriate SBI equivalents. The adapter supports an aggregate



VAX-11/780 I/O System

throughput rate of 1.5 million bytes per second.

High-performance mass storage devices—magnetic tapes and most disks—connect to a VAX-11/780 system through the MASSBUS. The MASSBUS adapter (MBA) links the MASSBUS to the SBI and performs mapping, buffering and control operations. A total of four MASSBUS adapters can be connected to a VAX-11/780 system, each permitting transfers at rates up to 2 million bytes per second to and from physical memory.

These bandwidths can easily accommodate the most demanding data throughput requirements. The following paragraphs discuss conditions and I/O techniques by which VAX-11 users easily can attain high throughput rates based on a required number of transfers per second.

Throughput Rates Up to 10,000 Transfers Per Second: For data rates up to 10,000 transfers per second, users may connect a simple interface directly to the VAX-11/780 UNIBUS. Even at this rate, the VAX-11/780 has spare processing capacity for other activities.

In this mode of operation, the user provides the necessary program control. The

VAX/VMS operating system makes I/O operations particularly easy for VAX-11 users through its Queue I/O system service routine.

Throughput Rates From 10,000 to 150,000 Transfers Per Second: At some point, the overhead that a stream of external interrupts imposes on the processor becomes very high. This situation is most notable whenever the processor spends the majority of its time responding to interrupts and little or no time processing. The actual point at which overload occurs depends on how much user applications must do between interrupts.

One way to relieve the processor of a heavy interrupt load is to transfer data directly to memory in blocks. This mode of I/O operation requires some form of block transfer or direct memory access (DMA) device, which must be able not only to handle the data transfer, but also to control the interface as though the interface were directly on the UNIBUS.

Microprocessor technology provides an effective technique for offloading the central processor. A microprocessor front-end can migrate some of the functions of the operating system into a microprocessor, which

manages the data transfers between a user device and main memory. This added capability does much to improve throughput, and at the same time, dramatically reduces processor overhead.

One example of microprocessor technology is the LPA, DIGITAL's high-throughput data acquisition front-end. The LPA is capable of transferring data at an aggregate rate of 150,000 transfers per second. For throughputs in excess of this, VAX-11 users can configure multiple microprocessor front-ends, or consider the I/O technique described next.

Throughput Rates Above 150,000

Transfers Per Second: For maximum I/O throughput, it is necessary to approach the throughput rates of the actual I/O buses. By connecting a digital I/O interface to one of the VAX-11 buses—UNIBUS or MASSBUS—a user, in principle, can effect data transfers at speeds approaching the bus bandwidths: 1.5M bytes in the case of the UNIBUS, and 2M bytes in the case of each of the four MASSBUSes.

The VAX-11/780 supports two digital I/O interfaces: the DR11 and the DR70-A. Both eliminate program controlled data transfers. The DR11 is a direct memory access interface through the VAX-11 UNIBUS. For even higher rates, the DR70-A supports block transfers using a direct-to-memory channel through the MASSBUS.

Interfaces and controllers such as these are able to optimize system performance because they have a high degree of built-in "intelligence." Microprocessor offloading, direct memory access and block transfer capabilities make I/O processing easier for VAX-11 users; the interfaces handle the most demanding performance needs for digital and analog I/O operations.

VAX System Responsiveness

System responsiveness is a measure of the elapsed time between the occurrence of an external stimulus and the system's meaningful reply.

For the computer system to be able to provide a timely response, it must be able to sense the stimulus, determine what the sti-



mulus means (the "process" in the above diagram), and reply in a useful way.

Usually, the stimulus takes the form of a change of voltage or contact closure, but it may also occur as the detection of a given bit pattern on a digital interface. Whatever its form, the real-time interface translates the stimulus into a system interrupt which, just as for programmed I/O, causes the processor to switch from its current activity to a new activity that handles the interrupt.

On the VAX-11, interrupt dispatching is extremely fast due to a sophisticated hardware priority system and the design of the device drivers.

The VAX-11 uses 32 hardware priority levels to accept and service interrupts. The upper 16 levels service the high priority hardware interrupts (device interrupts, for example), while the lower levels service interrupts generated by software.

To ensure that the VAX-11 processor is always ready to accept new device interrupts whenever they occur, the VAX/VMS operating system minimizes the amount of time it actually spends servicing interrupts at higher interrupt priority levels. For example, it performs activities like status reporting, event flag setting, and device deallocation at software interrupt priority levels. Since these less critical operations receive lower priority, they do not interfere with the servicing of additional interrupts.

The VAX/VMS device drivers also delay noncritical operations (like scheduling the application program) until after they have started a new I/O request. Thus, drivers can overlap I/O requests for better system throughput.

In addition, on the VAX-11, device interrupts are vectored to interrupt service routines; each device has its own vector value. This minimizes the time from the interrupt occurrence to the execution of the device dependent code.

This responsive interrupt system, plus a simple device driver, can service as many as 10,000 interrupts per second.

VAX Processing Power

System processing power involves the number of decisions that the computer system can perform per event, which, in turn, is measured by the complexity of the process to be performed on the real-time data.

The capability that is required to run a real-time process need not necessarily reside within the processor or in its operating system and user programs. In the case of the intelligent front-end, for example, it can be migrated away from the CPU and positioned where it is really needed, at the periphery. (In fact, the more powerful the processor, the greater the benefits of migrating special-purpose functionality outward in order to release the processor for its real purpose—that of computation.)

However, a system may contain many architectural characteristics that serve to improve its decision and performance capabilities. The VAX-11 supports several.

Powerful Instruction Set: Program performance gains on the VAX-11 can be attributed to its extensive and specialized instruction set.

Some instructions condense entire high-level language constructs into one or a few instructions for faster program execution. Others accelerate common mathematical operations.

Some instructions expedite system operations during real-time job scheduling and context switching. Others speed and simplify standard programming procedures; for example, what requires many move and store instructions on some systems takes only a single MOVC operation on the VAX-11.

Instructions that perform two- and three-operand arithmetic operations, integer and floating point arithmetic instructions, subroutine call and return instructions,

```
GOTO (10,20,30,40,50), I => CASEL I(R11),#1,#4
                             .DISPL .10
                             .DISPL .20
                             .DISPL .30
                             .DISPL .40
                             .DISPL .50
```

The computed GOTO sequence, used in FORTRAN programming, translates into the VAX-11 CASE instruction as shown above, illustrating the system's ability to translate entire high-level language constructs concisely and quickly.

packed decimal instructions, and a host of others—over 240 in all—make assembly language programming and high-level language execution extremely fast and efficient on the VAX-11.

Instruction Buffer: A special hardware cache, called the instruction buffer, speeds program execution. The instruction buffer allows the VAX-11 processor to fetch and decode the next program instruction while it executes the current instruction.

Floating Point Accelerator: An optional floating point accelerator significantly increases VAX-11 system performance for floating point operations.

The floating point accelerator is an independent processor that operates in parallel with the VAX-11 processor. The accelerator prefetches instructions and accesses main memory information. It then overrides the system's standard floating point microcode to execute floating point instructions much faster than the standard floating point microcode.

The following timings (single precision) are typical of the improvements that result with the FPA:*

Operation Reg. to Reg.	VAX-11/780 Standard	VAX-11/780 FPA
Floating Add/Subtract	2.2	0.8
Floating Multiply	5.2	1.2
Floating Divide	8.0	4.2

Writable Control Store: The VAX-11 supports a 12K byte writable control store, a special hardware feature allowing users to implement microcoded instructions of their choice. VAX-11 implementation tools and documentation will help users employ this feature to their best advantage.

Interprogram Communication and Control: VAX/VMS provides several techniques that make communication and synchronization among executing programs efficient and fast.

Applications can share data and code, for example. Global sections provide the means for programmers to indicate common data or common program code and to designate common communication regions in

*On the VAX-11, applications can always use floating point instructions, with or without the floating point accelerator. Using the floating point accelerator, however, improves performance significantly.

memory so that applications can easily and quickly exchange information, with reduced memory requirements.

Programs can also make use of a software data structure, called a "mailbox," to read and write data.

Users can define common event flags and then use them during programming to signal activities that have meaning to several programs collectively, such as the completion of a sample.

VAX/VMS even allows programs to control other programs. One application program can create or delete another. It can conditionally initiate and suspend execution of that program as events require. And it can use system timing services to schedule the execution of the program based on a specific time.

These are many of the ways that the special capabilities of VAX-11 architecture, and the VAX/VMS operating system, can improve system throughput and responsiveness during real-time processing.

Real-time Processing in a Virtual Memory Environment

The VAX-11 32-bit architecture supports unprecedented program address space—over four gigabytes (4×10^9) of virtual memory. Half of this—two gigabytes—is reserved for user programs and data. Two gigabytes of virtual memory gives VAX-11 users essentially unlimited programming capabilities.

VAX-11 users have a great deal of control over memory management operation. A main goal of the VAX-11 design was to provide users with a high-performance application environment. Thus, VAX-11 users can execute real-time applications with absolutely no paging or swapping activity at all. If they choose to utilize paging and swapping, they can directly influence the resulting overhead. This eliminates a traditional concern of users who have equated memory management operation with overhead on system response and program performance.

User Controls: VAX-11 users can define parameters that describe the general system environment. Many of the system pa-

rameters provided by the VAX/VMS operating system directly relate to memory management operation and program execution behavior. These parameters determine such system characteristics as whether any system paging will occur at all; if so, how many pages will be allowed in memory at one time; how many pages will be brought into memory with each disk access; and how long each job will be allowed to execute, to name a few.

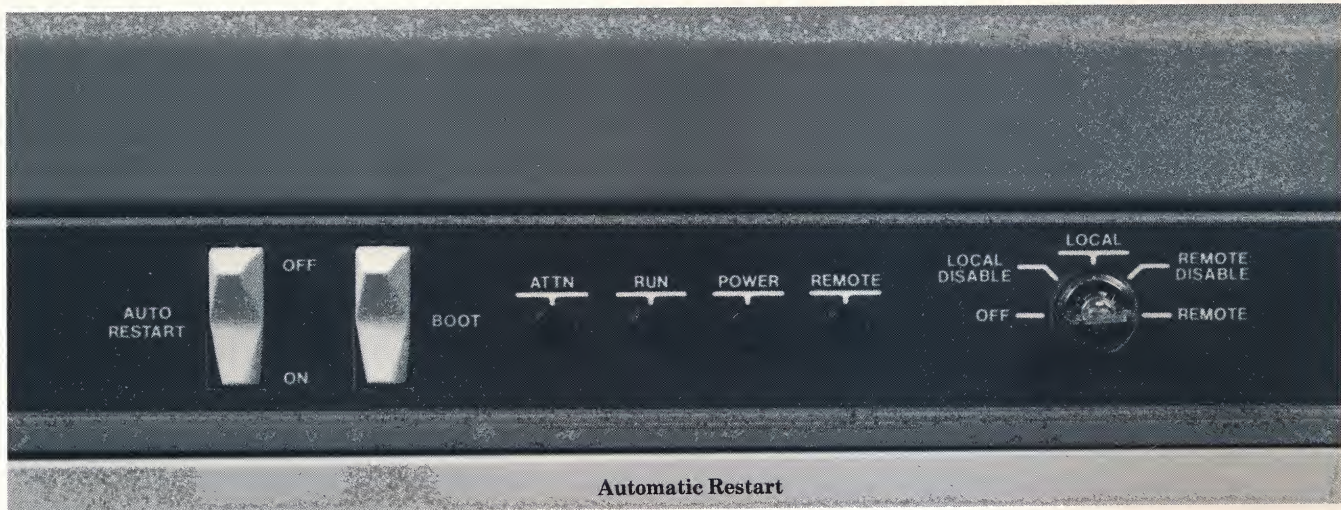
VAX-11 users can further control overhead by setting quotas and privileges on individual user applications. They can determine for each a specific program execution priority; a limit on the number of memory-resident pages; and exclusive or restricted access to a particular device, for example.

A privileged user application itself can control overhead by locking specific program pages in memory or by locking an entire program in memory.

FREE LIST: 933		PAGE MANAGEMENT STATISTICS 16:13:12				MODIFY LIST: 0	
NAME	VALUE	RATE /SEC	AVG RATE	NAME	VALUE	RATE /SEC	AVG RATE
PAGE FAULTS	211	159.84	28.55	FREE LIST	191	144.69	25.19
PAGES READ	10	7.57	1.81	MODIFIED LIST	7	5.30	0.97
READ I/Os	4	3.03	0.55	DEMAND ZERO	6	4.54	1.67
PAGES WRITTEN	48	36.36	1.67	WRITE IN PROG	3	2.27	0.10
WRITE I/Os	3	2.27	0.10	SYSTEM FAULTS	156	118.18	22.91

The VAX/VMS DISPLAY utility program helps users accurately refine system operation. Users observe system statistics such as memory management activity, I/O activity, system utilization, and other relevant information on a video terminal, then readjust system and program behavior for a particular application environment.

Optimized Paging Activity: The VAX/VMS paging algorithm also minimizes the overhead involved in paging operations. On the VAX-11, each program pages against itself—in order to make room for new pages, the operating system discards used pages from the executing program only, not from any other active jobs on the system. Consequently, the performance of real-time applications cannot be impaired by the paging activity of any other active jobs.



Job Scheduling By Priority: VAX-11 users can guarantee that real-time applications will receive favored, even exclusive, execution privileges and processor control.

The VAX/VMS operating system uses 32 levels of software priority for scheduling purposes, with the upper 16 levels reserved for real-time applications. Since the system scheduler always selects for execution the job having the highest priority that is ready to execute, users (with the proper privilege) can guarantee job scheduling by assigning their most critical applications priorities greater than 15. Once executing, a job always retains control until it finishes execution, enters a wait state, or is preempted by a job having higher priority.

Fast Context Switching: Whenever the VAX/VMS scheduler selects a new job for execution, the operating system handles the transition between the currently executing job and the new job, i.e., the context switch,* quickly and efficiently.

On the VAX-11, a context switch is handled by instructions created specifically for the purpose—Save Process Context and Load Process Context. VAX/VMS is able to save and restore context using a few instructions to do what takes entire software routines on other systems. In fact, a *complete* context switch on the VAX-11/780 takes only 180 microseconds.

*A context switch on the VAX-11 involves saving the context of one program and loading the context of another; context is *all* the information needed by the operating system to reschedule the two jobs. Often this term is defined as something much less, such as merely switching register sets.

Protecting the System Environment

Of importance to any user executing high performance and event-driven applications is assurance that the system will experience minimal downtime and that it will be adequately protected against misuse.

The VAX-11 supports a range of recovery and maintenance operations that promote easier system management and uninterrupted system service.

Its time-of-year clock permits fully automatic recovery from a power interruption or a hardware/software failure, which means no user or system management intervention is necessary to restart the system. VAX/VMS automatically determines if memory contents are valid following a failure, and if so, continues all I/O operations from their point of interruption and resumes execution of all running programs. If memory is not valid, the operating system is automatically rebooted from disk.

Capabilities for defining user privileges, setting priorities, managing the system configuration, and generally overseeing system usage give VAX-11 users complete control over the allocation of system resources, and complete confidence in the reliability of the VAX-11 system environment.

VAX Real-time Interfaces



The VAX-11/780 supports several interfaces that allow users to easily connect special application-specific I/O devices to the UNIBUS* and the MASSBUS.

In addition, two communication interfaces allow the formation of real-time networks comprising multiple VAX-11 and PDP-11 systems. These interfaces support local and remote networking features that are ideal for applications involving data collection, simulation, or process control.

Interfaces provided for use specifically in a real-time environment are described below. The "VAX-11/780 Systems and Options Summary" lists all VAX-11 hardware options.

LPA

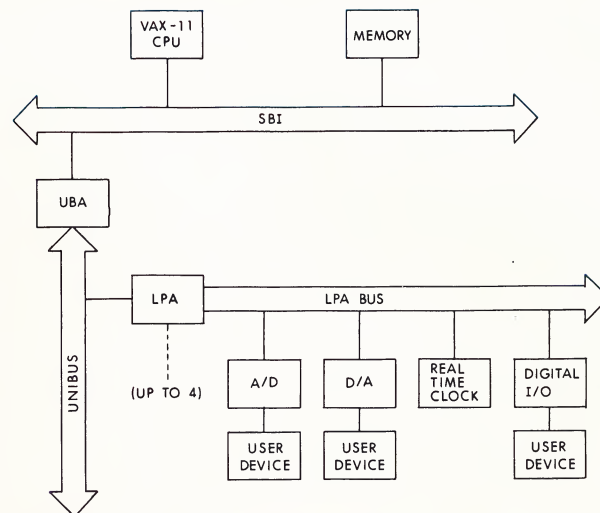
The LPA, an intelligent direct memory access controller for laboratory data acquisition interfaces, provides high functionality and high throughput real-time I/O on the VAX-11. The LPA uses a microprocessor subsystem that buffers and transfers real-time data collected by the I/O interfaces to and from main memory. This design allows data reduction and data acquisition operations to occur concurrently.

Since applications can now use the VAX-11 processor to process data rather than drive peripherals, system throughput is very high. For example, the LPA is capable of transferring data at an aggregate rate of over 150,000 samples per second (using two

*DIGITAL also supplies a number of aids for those users who wish to build their own interfaces to the UNIBUS.

analog-to-digital converters operating on two separate channels). As many as four LPAs can be connected to a single VAX-11/780 system through the UNIBUS.

The LPA currently supports analog-to-digital and digital-to-analog converters, digital I/O, and a real-time clock.

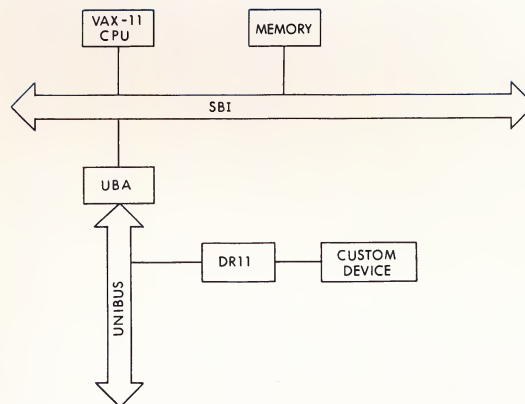


LPA Interface

DR11

The DR11 is a general-purpose, direct memory access interface that connects custom devices to a VAX-11/780 UNIBUS. Rather than requiring program controlled data transfers, the DR11 moves data directly between memory and the user device. For maximum flexibility, the throughput achieved with the DR11 is under full user control.

A combination of speed, high bandwidth through parallel lines, and direct memory access capability make this interface well suited for connecting array processors, graphics processors, special-purpose analog/digital converters, or any local real-time devices.

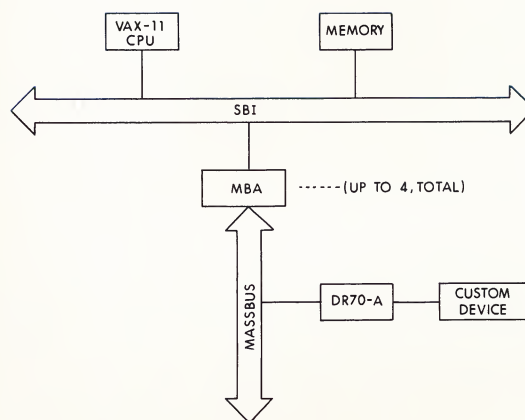


DR11 Interface

DR70-A

Similarly, the DR70-A is a general-purpose direct memory access interface that connects custom devices to a VAX-11/780 MASSBUS. The DR70-A allows block data transfers to be performed to or from memory; the transfer rate spans a range of 1.2 to 2M bytes per second, depending on system configuration and load.

Because the DR70-A can use a dedicated high-speed MASSBUS, it is used typically in applications requiring a higher speed interface than that provided by the DR11. In addition to high-speed throughput, the DR70-A offers high reliability because of end-to-end parity protection on all data transfers.



DR70-A Interface

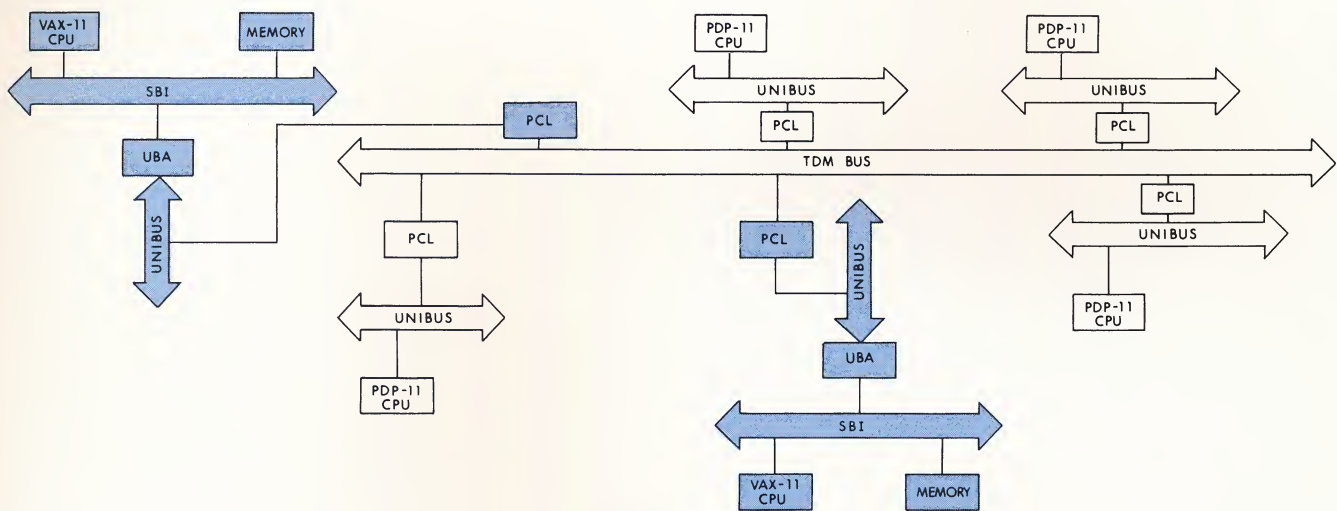
PCL

The PCL is a parallel communications link that connects multiple PDP-11 or VAX-11 processors in a local network. The network can consist of as many as 16 processors; each can communicate directly with any other processor in the network.

Communications occur in a direct memory access block mode over a time division

multiplexed (TDM) 16-bit parallel bus. The maximum TDM bus bandwidth is 8 million bits per second.

The PCL allows any processor or other interface in the network to be powered on or off, and additional processors to be incorporated into the network, without disrupting the rest of the network.



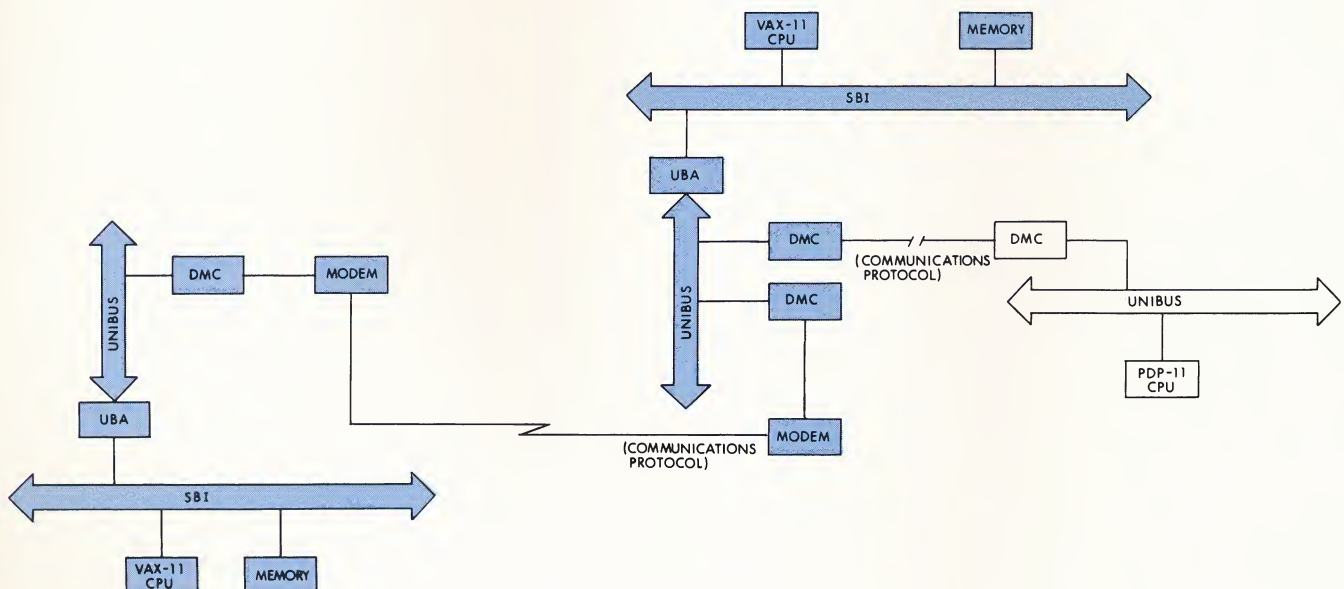
PCL Interface

DMC

The DMC Network Link is DIGITAL's synchronous serial UNIBUS interface for local and long distance (remote) networks. The link uses DIGITAL's standard DECnet communications protocol, called DDCMP; it

supports direct memory access transfers, and both full- and half-duplex operation.

DMCs can be configured for local operation at speeds up to 1 million bits per second, and remote operation at speeds up to 19,200 bits per second.



DMC Interface



VAX Application Development Tools

DIGITAL systems are designed for users at all levels of experience. Operating systems are interactive and easy to learn. Languages and application development aids make it possible for users to concentrate on developing applications, not on understanding system usage procedures. In keeping with these standards, VAX/VMS is DIGITAL's most user-oriented and complete operating system yet.

Programming Languages

VAX/VMS native programming languages make full use of the 32-bit features of the VAX-11 architecture for compact source language translation and fast program execution. The compilers for these languages produce code that can be shared by users for more economical memory utilization. In addition, all VAX/VMS languages support standard calling sequences. This capability allows application programs to easily communicate among themselves and gives high-level languages the ability to access VAX-11 features.

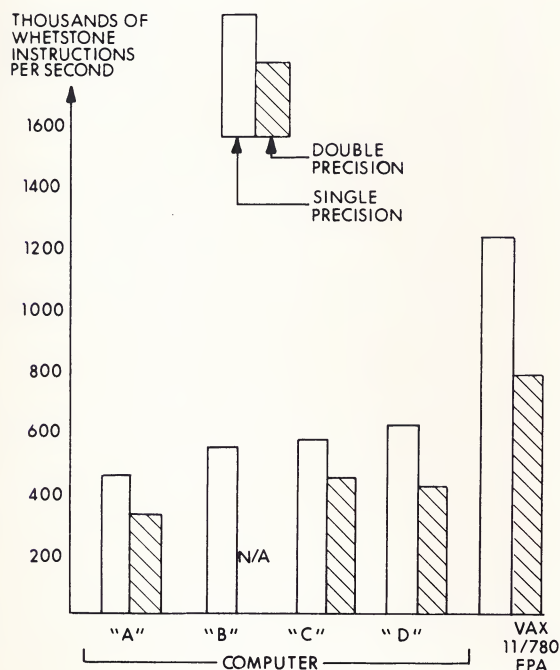
VAX-11 FORTRAN IV-PLUS:

FORTRAN has long been regarded as *the* language for scientific programming; its language statements are well suited for applications that use standard mathematical expressions and equations and that call for extensive computation.

VAX-11 FORTRAN IV-PLUS is DIGITAL's most sophisticated FORTRAN compiler. It conforms in all respects to the ANSI standard X3.9-1966, and draws from the proposed ANSI-77 standard by incorpo-

rating the CHARACTER data type and block IF statements.

The FORTRAN IV-PLUS compiler uses the VAX-11's floating point and character string instructions, and other special instructions, to produce highly optimized object code for fast program execution. In fact, Whetstone Benchmark* results show that FORTRAN IV-PLUS programs execute significantly faster on the VAX-11/780 than on any other computer in its class.



Program performance for machines A through D (all 32-bit or larger machines) was measured by other vendors in competitive situations.

*The Whetstone Benchmark uses a mix of FORTRAN statements that reflects the frequency of statement usage measured in 900 actual programs. The higher the number of Whetstone instructions per second, the better.

VAX-11 MACRO: VAX-11 MACRO is the system's 32-bit assembly language. It gives programmers the use of over 240 machine level instructions for creating extremely efficient and precise programming structures.

The language provides a number of assembler directives for complete control of listing output, message displays, data storage formats, and other assembly functions.

In addition, the language's extensive macro capability reduces the amount of source code and time required to create programs. Programmers can define individual macros that describe entire sequences of operations. The macro definition is required only once, but the operations subsequently can be used in any program, through a single macro call.

VAX-11 MACRO is quite similar to DIGITAL's PDP-11 (16-bit) MACRO language. Instruction formats, instruction mnemonics, and addressing modes are very similar. VAX-11 users who are already familiar with PDP-11 programming find the transition to VAX-11 programming easy.

VAX-11 BLISS-32: VAX-11 BLISS-32 is a high-level systems implementation language designed to simplify development of real-time programs, operating system software, and compilers.

BLISS-32 allows direct access to memory addresses, access to general purpose registers, access to system hardware, macro support, library facilities, and other features and operations which would normally require machine language programming.

BLISS-32 supports structured programming concepts to promote easier program development and maintenance.

Application Tools

VAX/VMS supports a variety of application aids that can considerably shorten the program development cycle. System utilities let users execute programs quickly, interactively examine and evaluate the results of program execution, and modify and fine-tune programs and data on-line. System networking capabilities allow several systems to share resources.

An extensive documentation set describes all aspects of the system and includes aids for users who want to write drivers and diagnostic software for custom devices.

System Utilities: VAX/VMS utility programs provide the means for users to create and edit source program files and data files, share programs and routines, and perform general system activities.

Standard utilities include several editors for source program creation and modification; a linker for source program translation; standard libraries and a librarian utility for library maintenance; and record and file management utilities.

RSX-11M compatible development aids allow users to develop applications for DIGITAL's RSX-based operating systems.

```
Username: JCR
Password:
Welcome to VAX/VMS Version 1.01

$ TYPE
$_File: FORT.COM

$ON ERROR THEN GOTO END
$TEMP := 'P2' !DOES COMMAND LINE CONTAIN PARAMETER?
$IF TEMP .EQS. "" THEN GOTO REGULAR
$FORTRAN 'P1/'P2 !COMPILE WITH OPTIONAL PARAMETER
$GOTO CONTINUE
$REGULAR: FORTRAN 'P1 !OR COMPILE WITHOUT PARAMETER
$CONTINUE: PRINT/DELETE 'P1.LIS !PRINT LISTING
$LINK 'P1 !LINK PROGRAM
$SHOW STATUS
$RUN 'P1 !RUN PROGRAM
$END: SHOW STATUS !DISPLAY EXECUTION STATISTICS
$ @FORT GRAPH/DEBUG=TRACEBACK
```

The VAX/VMS system/user interface is the standard DIGITAL command language, DCL. English command verbs, system prompts, and other features of the language make use of the operating system and the utilities interactive and easy. Here the user types a command to display on the terminal the contents of a command procedure file and then uses that command procedure to initiate execution of a FORTRAN program.

Thus, users can develop and debug PDP-11 programs on a VAX-11 for migration between systems or for downline system loading to an RSX-11S site.

Program Debugging: VAX/VMS program debugging capabilities are extensive. Symbolic traceback causes the VAX/VMS operating system to trace the source of an execution error through all calling modules and print module names and line numbers for programmer reference. The symbolic debugger utility enables comprehensive program debugging. It simplifies interactive program execution and control by allowing users to reference addresses using the symbolic names of variables and locations as they appear in the actual source program.



Data Maintenance: For easy on-line data management and maintenance, VAX/VMS provides an inquiry language and report writer called DATATRIEVE. Simple commands let users find, print, update and sort data records, easily and interactively at the terminal.

Distributed Real-time Processing:

Networks comprising VAX-11 and other DIGITAL systems may be formed using either the PCL parallel communications link or the DMC synchronous serial interface link. Typically, the PCL is used for high-speed direct communication between local networks, while the DMC is used for versatile remote communication.

DECnet-VAX is DIGITAL's standard networking capability for VAX-11 systems connected by DMC communications lines. Nodes in such a network can share resources in a variety of ways.

A program executing under the control of one system in a VAX-11 network, for example, can communicate with a cooperating program at an adjacent site using standard communication protocols. The communication methods used in the networking system are not restricted to any particular operating system, so no special considerations limit interacting programs.

VAX-11 users can transfer files from one system to another in the network using standard system commands. Users can access devices located at other systems, subject to any restrictions of the target system, and can transfer and initiate command procedures between systems.

Finally, users can create applications for DIGITAL's RSX-11S (application-specific/execution-only) systems using standard VAX/VMS program development aids, then downline load the application system from the VAX-11 system for direct execution at the RSX-11S site.

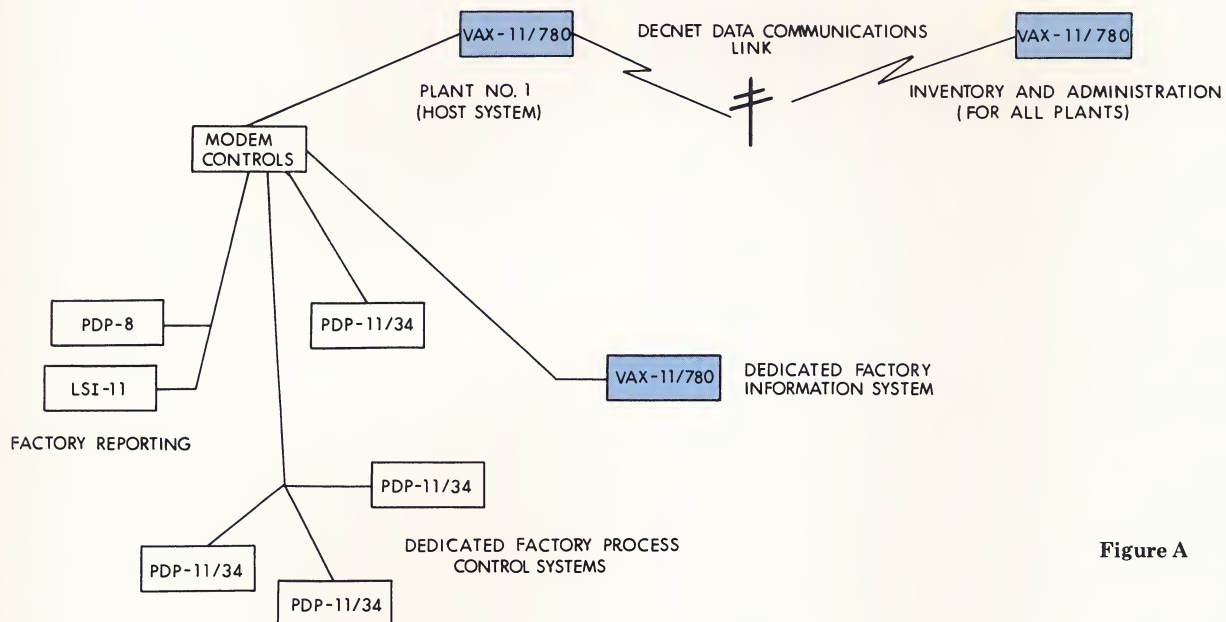


Figure A

Figure A shows a host VAX-11/780 system connected to a series of VAX-11, PDP-11, and PDP-8 minicomputers located at various sites on a manufacturing production floor. The computers are placed strategically for operator-controlled data entry and information gathering and special-purpose I/O operations. Data is sent to the host VAX-11 for analysis and processing and then back to the special control peripherals located on the floor; at the same time, general inventory and accounting information gets sent to a second VAX-11 system located at the main administrative office cities away.

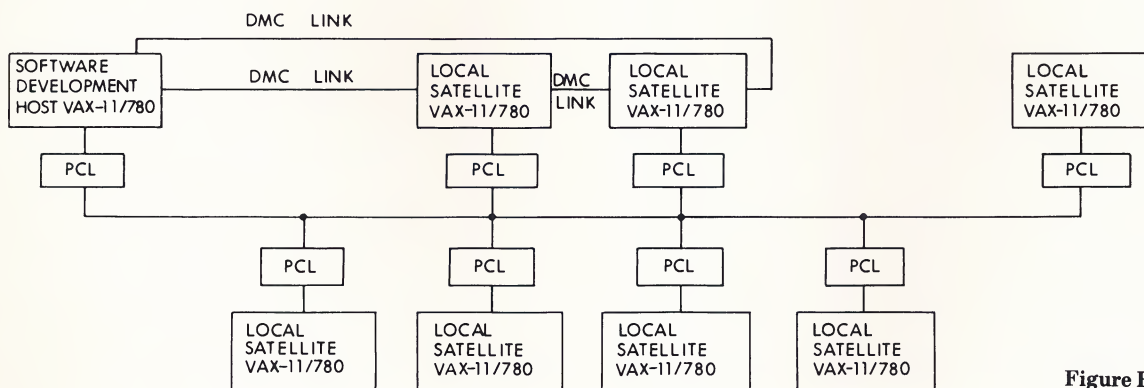
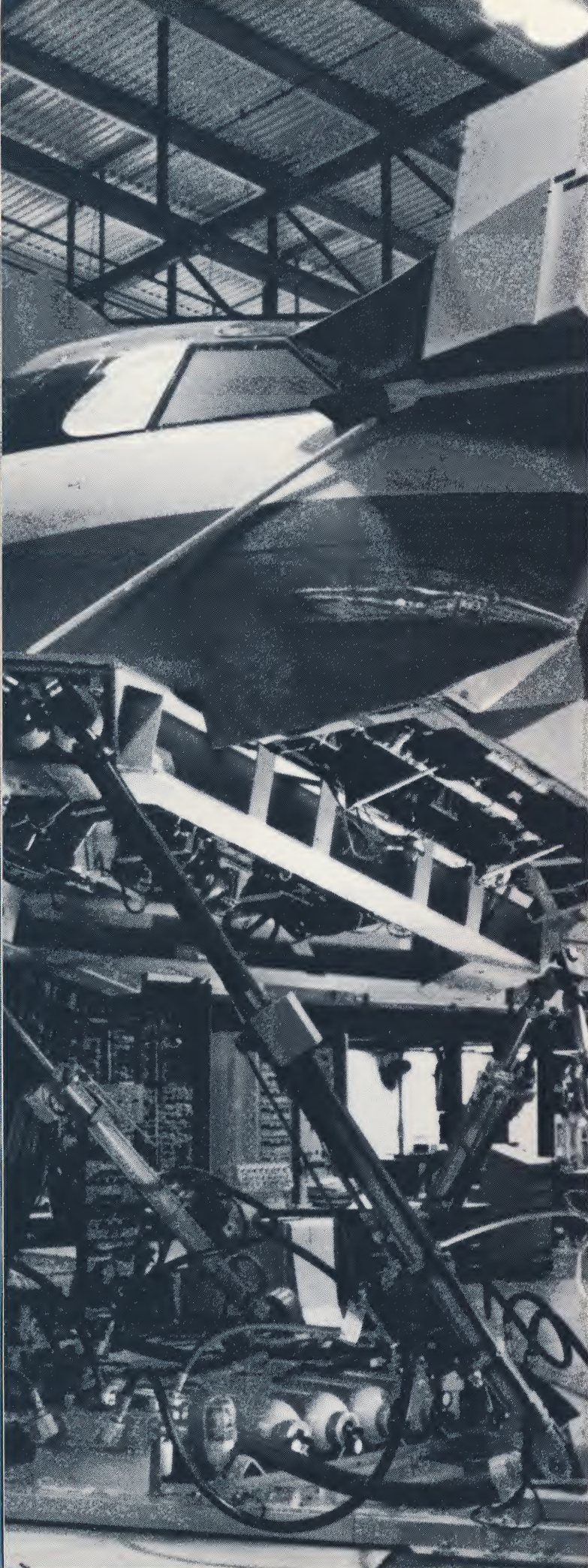


Figure B

Figure B shows a different type of real-time network in which VAX-11/780 systems perform sophisticated radar and missile simulation. Eight VAX-11 systems share the complex computation required by the application and are connected locally by PCL links for high-speed parallel communications. One of these is also used for program development and is connected to several of the others by DMC lines for versatile DECnet communication. (This application is further described in the application section of this overview.)



VAX Real-time Applications

The following application stories illustrate how customers are already using VAX-11 systems to address real-time application needs. The stories are representative of a cross section of user experiences.

Aircraft Flight Simulation —a Demanding Job in Real-time

Commercial airlines and military aviation are turning more and more frequently to computerized flight simulators to give their student pilots realistic flight training on the ground. Many simulations require turn-around times (data sampling, computation, and output) in the 25 to 50 millisecond range.

Flight simulator applications use a series of real-time linkages to interface a computer system with an airplane cockpit mounted on a motion platform, and with an instructor's station. The student pilot sits in the cockpit and operates aircraft controls, reacting to staged stimuli as though he were participating in a real-life flight situation. An instructor sits in the instruction station and monitors the student's training session.

The instructor typically uses a graphics display terminal to visually monitor and control the student's progress. He interactively inputs commands to introduce situations which the student might conceivably face—an engine failure or a change in weather conditions, for example. Cockpit motion and visual systems within the cockpit respond in real-time to the instructor's control commands, and to the pilot's responses, creating an extremely realistic flight environment.

A supplier of flight simulators in Canada needed a computer system for a new family of commercial airline wide-body simulators. This supplier had formerly employed a

16-bit minicomputer in their simulators, but felt that because modern wide-body aircraft contain more complex subsystems, the new simulators would require more throughput than the 16-bit minicomputer could provide.

They realized that a basic aircraft design might serve the industry for twenty years or more. But significant modifications to navigational aids, engines, control systems—virtually every other aspect of the airplane—would be continually introduced on an ongoing basis. Each new change implemented in the actual aircraft would require a corresponding change in the simulators. Consequently, software development capabilities were an important consideration in their selection of a system.

In addition, since their customers would be airlines located throughout the world, they were concerned about the availability of computer service and spares.

The VAX Solution: This flight simulator supplier found DIGITAL's VAX-11/780 to be the best choice for their new family of wide-body simulators. First of all, they were convinced that the VAX-11 could easily handle the demanding requirements of their application. Its system response and throughput were demonstrated as more than adequate for transmitting the large number of discrete and digital signals through their simulator linkages. Computational performance easily accommodated the "number crunching" necessary to solve the complex flight equations which allowed the system to generate appropriate and credible behavior responses.

They found they could use VAX/VMS's extensive interactive program development features, and its FORTRAN IV-PLUS language, to write high performance application and graphics software; they no longer had to use assembly languages to effect the program performance required. They were able to design their application and incorporate aircraft modifications as they occurred, quickly and easily, with minimum maintenance and update costs.

They were completely convinced when they found that DIGITAL's Field Service technicians and extensive spare parts organization could provide 24-hour worldwide on-site support.

Defense Simulation—Using a Sophisticated Real-time Network for a Complex Job

A major system house, under government contract, wanted to construct a network of computer systems to handle large-scale problems in radar and missile simulation. The real-time part of their job involved analyzing real or simulated radar blips to determine those representing missiles, and then simulating an interception to the approaching missiles.

This task required high-speed computation, a large address space, and the ability to handle programs which had grown to more than 3 million bytes in size.

Since the customer also intended to use the system to develop their simulation programs, they required a highly interactive computer with good development tools.

The VAX Solution: The large address space requirement eliminated 16-bit computers from consideration, and after thorough investigation of wide-word machines, this customer chose 10 VAX-11/780s.

While sophisticated in its capabilities, this particular network was conceptually simple, and the VAX-11 systems far exceeded performance expectations. Strong interconnect capabilities were a prime reason for their selection.

Eight of the systems were connected in a combination of PCL and DMC links. The PCL links enabled systems to share the heavy computational load and to solve complex simulation algorithms in real-time, while the DMC links allowed versatile software development to occur at the same time.

The remaining two systems were used in a stand-alone mode to offload another mainframe and for independent program development and research.

In addition to computational load-sharing, the network gave the program developers a highly interactive and available resource for designing software. In fact, the VAX-11 network became extremely popular with the programmers because it provided faster turnaround than did their former mainframe. If one system was fully loaded, the network automatically rerouted develop-

ment to another VAX-11, with no noticeable effect on response time.

The VAX-11/780 originally beat the competition because of price/performance and large address space considerations. But its interactive nature and excellent development tools expanded its usefulness well beyond original plans.

Industrial Process Control—Stepping up Production with a High-throughput System

A large manufacturing company had used a 16-bit minicomputer to automate the use of complex pattern information in one of their production operations. Every run required a setup time in which several partial patterns, each consisting of over 100 megabytes of data, needed to be mixed selectively in order to create a final pattern. This was communicated to a number of satellite computers that controlled the finishing stages of their production process.

The time needed to set up a pattern was primarily dependent on three things: the time it took to load the required pattern information into computer memory from disk, the size of the internal buffer space available for loading and processing (i.e., mixing) the

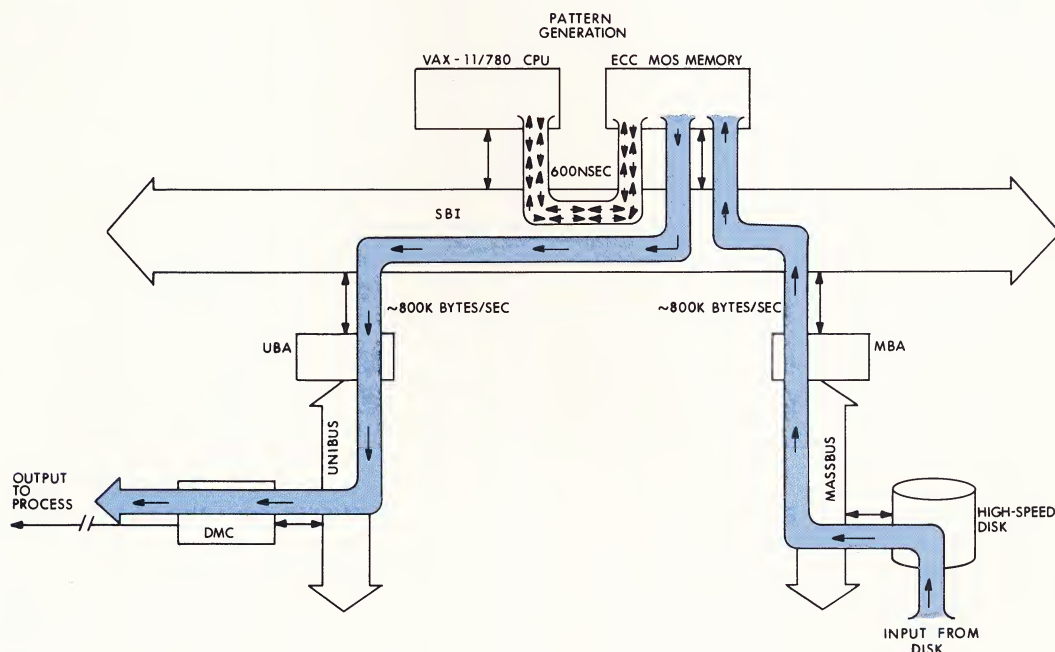
composite pattern information (determined by the computer's addressing capabilities—64K bytes on this machine), and the time needed for communicating the final pattern to the satellites.

Because there were several setups during a shift, and because the process was idle while each setup was occurring, the company wanted to reduce the setup time as much as possible. The problem was how.

The VAX Solution: A VAX-11/780 turned out to be the ideal solution. Its four MASSBUS adapters allowed data to be fed rapidly from four disk channels into memory simultaneously. Its 32-bit word length supported an addressing capability far exceeding the 64K bytes allowed by the smaller machine, with the result that much more pattern data could be loaded and processed much faster.

In addition, the direct memory access capability provided by their system's DMC communications interface meant that the final pattern could be communicated rapidly to the satellite processors through a high-speed communications link.

Not only did a VAX-11/780 significantly reduce total setup time for this company by processing larger blocks of information and providing faster I/O throughput, it delivered the high standards of reliability required by their application.



Laboratory Data Acquisition—Keeping up with High-energy Physics Research

A national government-funded research organization provides the facilities for qualified scientists to perform experiments in particle physics research. Experiments involve injecting high-energy electrons and positrons, which originate in a linear accelerator, in opposite directions into a hexagonal shaped vacuum storage ring. The electrons and positrons collide, to be replaced by pure electro-magnetic energy which converts into sub-nuclear particles that the scientists can study.

For a proposed new series of experiments, the research facility wanted to provide appropriate particle detectors and associated computer systems. They designed a new 750 meter (2450 foot) diameter vacuum storage ring, which would incorporate six particle detectors, each with its own on-line computer system, to be located in experimental areas midway on each side of the storage ring.

In principal, each detector system would be responsible for monitoring particle collisions (or "events") within the vacuum storage ring. Events would occur randomly with time and with a frequency varying from one every two seconds to three per second. Data would be collected from some 10,000 channels, all connected to the central processor via an intelligent controller. A throughput of up to 40,000K bytes per event was required,

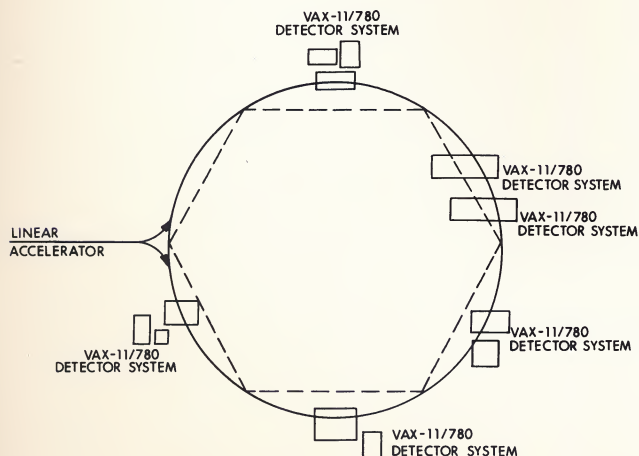
and the system had to be responsive enough to act on a decision as to whether data derived from an event would be of interest or should be discarded.

In terms of functionality, each system would be required to perform on-line computation concurrently with the data acquisition task on approximately 10% of all events, displaying the results on five interactive display terminals.

The VAX Solution: The research facility selected six VAX-11/780s for their on-line detector systems. Benchmarks showed that the VAX-11/780 met their real-time performance and data throughput requirements better than any of the other systems they were considering, and at much less total cost.

The one megabyte memory configuration with one 67 megabyte disk and one 176 megabyte disk running under the VAX/VMS operating system provided the environment required for the data acquisition and data analysis software. VAX/VMS's ANSI standard FORTRAN IV-PLUS provided a convenient migration path from existing software already in use at the central site. And the multi-user VAX/VMS operating system allowed new program development to take place concurrently with data acquisition.

The raw data collected, consisting of electrical signals representing time, position, velocity, and energy associated with the newly created particles occurring as the result of a collision, was read directly onto 1600 bpi magnetic tape. Data acquisition was handled by a CAMAC I/O system interfaced to the VAX-11/780 UNIBUS.



digital

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, Massachusetts 01754, Telephone (617) 897-5111 — SALES AND SERVICE OFFICES:
UNITED STATES — ALABAMA, Birmingham, Huntsville • ARIZONA, Phoenix, Tucson • CALIFORNIA, El Segundo, Oakland, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Santa Clara • COLORADO, Colorado Springs, Denver • CONNECTICUT, Fairfield, Meriden • FLORIDA, Miami, Orlando, Tampa • GEORGIA, Atlanta • HAWAII, Honolulu • ILLINOIS, Chicago, Peoria, Rolling Meadows • INDIANA, Indianapolis • IOWA, Bettendorf • KENTUCKY, Louisville • LOUISIANA, New Orleans • MARYLAND, Baltimore, Odenton • MASSACHUSETTS, Boston, Springfield, Waltham • MICHIGAN, Detroit • MINNESOTA, Minneapolis • MISSOURI, Kansas City, St. Louis • NEBRASKA, Omaha • NEW HAMPSHIRE, Manchester • NEW JERSEY, Cherry Hill, Fairfield, Princeton, Somerset • NEW MEXICO, Albuquerque, Los Alamos • NEW YORK, Albany, Buffalo, Long Island, Manhattan, Rochester, Syracuse, Westchester • NORTH CAROLINA, Chapel Hill, Charlotte • OHIO, Cincinnati, Cleveland, Columbus, Dayton • OKLAHOMA, Tulsa • OREGON, Portland • PENNSYLVANIA, Allentown, Harrisburg, Philadelphia, Pittsburgh • RHODE ISLAND, Providence • SOUTH CAROLINA, Columbia • TENNESSEE, Knoxville, Nashville • TEXAS, Austin, Dallas, El Paso, Houston • UTAH, Salt Lake City • VERMONT, Burlington • VIRGINIA, Richmond • WASHINGTON, Seattle • WASHINGTON, D.C. • WEST VIRGINIA, Charleston • WISCONSIN, Milwaukee • INTERNATIONAL — ARGENTINA, Buenos Aires • AUSTRALIA, Adelaide, Brisbane, Canberra, Darwin, Hobart, Melbourne, Perth, Sydney, Tasmania, Townsville • AUSTRIA, Vienna • BELGIUM, Brussels • BOLIVIA, La Paz • BRAZIL, Rio de Janeiro, Sao Paulo • CANADA, Calgary, Edmonton, Halifax, Kanata, London, Montreal, Ottawa, Quebec City, Toronto, Vancouver, Winnipeg • CHILE, Santiago • DENMARK, Copenhagen • EGYPT, Cairo • FINLAND, Helsinki • FRANCE, Lyon, Marseille, Paris, Puteaux • HONG KONG • INDIA, Bombay • IRAN, Tehran • IRELAND, Dublin • ISRAEL, Tel Aviv • ITALY, Milan, Rome, Turin • JAPAN, Osaka, Tokyo • MEXICO, Mexico City • NETHERLANDS, Amsterdam, Hague, Utrecht • NEW ZEALAND, Auckland, Christchurch, Wellington • NORTHERN IRELAND, Belfast • NORWAY, Oslo • PERU, Lima • PUERTO RICO, San Juan • SAUDI ARABIA, Jeddah • SCOTLAND, Edinburgh • SINGAPORE • SOUTH KOREA, Seoul • SPAIN, Madrid • SWEDEN, Gothenburg, Stockholm • SWITZERLAND, Geneva, Zurich • TAIWAN, Taoyuan • UNITED KINGDOM, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Reading, Welwyn • VENEZUELA, Caracas • WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nurnberg, Stuttgart • YUGOSLAVIA, Ljubljana •